

# Embedding Graphs into the Extended Grid

Michael Coury \*

February 7, 2008

## Abstract

Let  $G = (V, E)$  be an arbitrary undirected source graph to be embedded in a target graph  $EM$ , the extended grid with vertices on integer grid points and edges to nearest and next-nearest neighbours. We present an algorithm showing how to embed  $G$  into  $EM$  in both time and space  $O(|V|^2)$  using the new notions of *islands* and *bridges*. An island is a connected subgraph in the target graph which is mapped from exactly one vertex in the source graph while a bridge is an edge between two islands which is mapped from exactly one edge in the source graph. This work is motivated by real industrial applications in the field of quantum computing and a need to efficiently embed source graphs in the extended grid.

## 1 Introduction

In this paper, we describe a method for embedding any undirected source graph into the extended grid. We also introduce the concept of islands and bridges. This embedding problem is of interest theoretically, and it has real industrial applications in the field of quantum computing that motivate this research. We introduce a constructive algorithm to embed complete graphs in  $O(n^2)$ , thereby providing an upper bound.

An adiabatic quantum computer, such as one based on the system described by Amin, et alia [1], can be considered to be a graph. This allows for the computer to be programmed by formulating a given problem as a graph theoretic problem (e.g. cellular base station placement is formulated as maximum independent set) and then embedding the problem graph onto the graph representation of the quantum computer. Therefore, the problem of embedding a source graph (representing the problem) into a target graph (representing the quantum computer architecture) is both interesting and important.

In collaboration with David Grant and William Macready, we developed the idea of a connected subgraph in the target graph to represent a single vertex in the source graph as well as a randomized  $O(n^k)$ ,  $k > 2$ , and then a  $2n \times 2n$ , algorithm for embedding. Garey, Johnson, and So [2] consider connected subgraphs of vertices, called nets, in the context of circuit testing; that is, looking for short circuits. Opatrny and Sotteau [3] and Lin, et alia [4], describe, in the contexts of VLSI and parallel computing, embedding complete binary trees into  $EM$  using edge subdivision with vertex congestion of one, while Sang and Sudborough [5] use vertex congestion and contraction to embed larger meshes into smaller ones. Fraysseix, et alia [6] and Schnyder [7] describe  $O(n^2)$  algorithms to obtain Fáry embeddings.

### 1.1 Definitions

We define the extended grid ( $EM[m, n]$ ) to be an  $m$  row by  $n$  column lattice of grid points where every grid point has a potential edge to its immediate, or nearest, neighbours as well as to its next-nearest neighbours. We will abbreviate  $EM[m, n]$  as  $EM$ . More specifically,  $EM$  consists of the set of vertices given by the grid coordinates  $\{(x, y) | 1 \leq x \leq m, 1 \leq y \leq n\}$ , and for a vertex  $v = (x, y)$ , the edges incident on  $v$  are  $\{(u, v) | u \in (x, y \pm 1), (x \pm 1, y), (x \pm 1, y \pm 1)\}$ .

Given an arbitrary source graph  $G = (V, E)$ , the problem is to embed this graph as a subgraph of  $EM$ ,  $\hat{G} = (\hat{V}, \hat{E}, \hat{C})$ , where each  $v \in V$  maps to a nonempty set  $\vec{v} \subseteq \hat{V}$ , each edge  $(u, v) \in E$  maps to an edge  $(u', v') \in \hat{E}$  with  $u', v' \in \hat{V}$ ,  $u' \in \vec{u}$ ,  $v' \in \vec{v}$ , and  $\vec{v}$  is a connected subgraph whose edges are in

---

\*School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, V5A 1S6 (mcoury@cs.sfu.ca)

$\hat{C}$ . That is, each vertex in the source graph maps to a set of vertices in the target graph that form a connected subgraph. We call such a connected subgraph an island. The edges in the island are in the edge set  $\hat{C}$ . Each edge in the source graph also maps to an edge, which we will refer to as a *bridge*, in the target graph's edge set  $\hat{E}$  between two appropriate islands.

Put another way, let us define an island  $I_v$  to be a connected subgraph of the embedding  $\hat{G}$  where the vertices of  $I_v$  collectively represent the vertex  $v \in G$ ; i.e, the vertices of  $I_v$  are  $\vec{v}$ . Thus, we have a function  $\eta_V : V \rightarrow \hat{V} \times \hat{V} \times \dots \times \hat{V}$  that maps vertices to islands. We then define a bridge to be an edge  $(I_{u,s}, I_{v,t})$  connecting two islands  $I_u$  and  $I_v$ , given by the bijective function  $\eta_E : E \rightarrow \hat{E}$  that takes edges to bridges.

Given an input graph  $G = (V, E)$ , let us define an embedding of that graph to be a new graph  $\hat{G} = (\hat{V}, \hat{E}, \hat{C})$ . Also, let  $l$  be a labelling function that maps a vertex of an island in the embedding to vertices in the input graph,  $l : \hat{V} \rightarrow V$ . We can then say an island is

$$\hat{V}_v = \{v' | l(v') = v\}$$

and an edge set is

$$\hat{C}_v = \forall u' \in \hat{V}_v \exists v' \in \hat{V}_v, \{(u', v') | l(u') = l(v') = v\}.$$

From these definitions we get

$$\hat{V} = \bigcup_{v \in V} \{\hat{V}_v\} \quad , \quad \hat{C} = \bigcup_{v \in V} \{\hat{C}_v\}.$$

Furthermore, we define

$$\hat{E}_{uv} = \{(u', v') | l(u') = u, l(v') = v\}$$

and we get

$$\hat{E} = \bigcup_{(u,v) \in E} \{\hat{E}_{uv}\}.$$

Finally, we say  $I_v = (\hat{V}_v, \hat{E}_v)$ .

Thus, we create an embedding  $\hat{G}$  of the input graph  $G$  such that each vertex  $v \in V$  corresponds to an island,  $\hat{V}_v \in \hat{V}$ , where  $|\hat{V}_v| \geq 1$ , with a set of edges  $\hat{C}_v$  such that the graph  $\hat{G}_v = (\hat{V}_v, \hat{C}_v)$  is connected. Furthermore, for every edge  $(u, v) \in E$  there exists an edge  $(u', v') \in \hat{E}$  such that  $l(u') = u, l(v') = v$ .

## 2 Algorithm

We now provide an  $O(n^2)$  algorithm which will show how to embed any complete graph into the extended grid  $EM[n-1, n]$  using islands and bridges, where  $n = |V|$ . The embedding technique we describe will be call a braiding since we braid the layout of the islands together in the embedding. One way to think of it is as a series of swaps, whereby we maximize the number of swaps, and the number of new edges introduced, at each row transition.

The algorithm is as follows. For each odd-labelled vertex, we grow that island toward the right until it hits the boundary; at this point, the island reflects after a delay. Even-labelled vertices are treated similarly except the initial growth is towards the left. (Note that this orientation is arbitrarily chosen.)

We iterate through the rows, starting from the first. Given a numbered ordering  $n_V = \{1, 2, \dots, n\}$  of  $V$ , we layout all the vertices in ascending order on the first row. We proceed to layout the remaining rows as follows:

Let  $c_i(v)$  denote the column in row  $i$  for vertex  $v$ . For each  $v \in V$  and for each row, if  $n_V(v)$  is odd, then  $c_i(v) = c_{i-1}(v) - 1$  while  $c_i(v) > 0$ . If  $c_{i-1}(v) = 1$ , then let  $c_i(v) = 1$  and let  $j = i$ . For all subsequent  $i > j$ ,  $c_i(v) = c_{i-1}(v) + 1$ . If  $n_V(v)$  is even, the layout is reversed.

Thus,

$$c_i(v) = \begin{cases} n_V(v) + h - \lfloor \frac{n_V(v)+h-1}{n} \rfloor (2r_o + 1) & \text{if } n_V(v) \text{ is odd} \\ n_V(v) - h + \lceil \frac{-(n_V(v)-h-1)}{n} \rceil (2r_e + 1) & \text{if } n_V(v) \text{ is even} \end{cases}$$

where

$$\begin{aligned} h &= i - 1, \\ r_o &= h - (n - n_V(v)) - 1, \\ r_e &= h - n_V(v). \end{aligned}$$

---

**Algorithm 1** Braiding algorithm

---

**Input:** A graph  $G = (V, E)$ **Output:** An embedding  $\hat{G} = (\hat{V}, \hat{E}, \hat{C})$ 

```
 $m = |V| - 1$ 
 $n = |V|$ 
for  $nv = 1$  to  $n$  do
   $i = 1$ 
   $j = nv$ 
  if isOdd( $nv$ ) then
     $left = \text{true}$ 
  else
     $left = \text{false}$ 
  end if
  while  $i \leq m$  do
     $\hat{V}(i, j) = nv$ 
     $jlast = j$ 
     $i = i + 1$ 
    if  $left$  then
       $j = j + 1$ 
    else
       $j = j - 1$ 
    end if
    if  $j > n \wedge left$  then
       $j = n$ 
       $left = \text{false}$ 
    else if  $j < 1 \wedge \neg left$  then
       $j = 1$ 
       $left = \text{true}$ 
    end if
    if  $i \leq m$  then
       $\hat{C}(\text{toIndex}(i - 1, jlast), \text{toIndex}(i, j)) = 1$ 
    end if
  end while
end for
for  $i = 1$  to  $m$  do
   $uv = \hat{V}(i, \{1 \dots n\})$ 
  for  $j = 1$  to  $|uv| - 1$  do
    if  $E(uv(j), uv(j + 1)) \neq 0$  then
       $\hat{E}(\text{toIndex}(i, j), \text{toIndex}(i, j + 1)) = 1$ 
       $\hat{E}(\text{toIndex}(i, j + 1), \text{toIndex}(i, j)) = 1$ 
       $E(uv(j), uv(j + 1)) = 0$ 
       $E(uv(j + 1), uv(j)) = 0$ 
    end if
  end for
end for
```

---

---

**Algorithm 2** toIndex function

---

**Input:** the row  $r$  and the column  $c$ **Output:** the index,  $i$  $i = (r - 1) \cdot n + c$ 

---

The algorithm, shown in Algorithm 1, is quite simple and creates the braided embedding, as shown for  $K_6$  and  $K_{3,3}$  in Figure 1, and is scalable to any number of vertices. It is apparent that this is sufficiently scalable to embed any complete graph and therefore any graph.

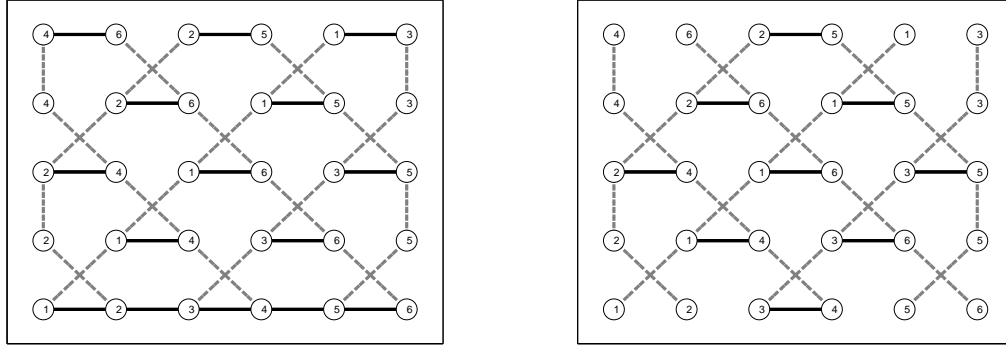


Figure 1: Braided embedding of (a)  $K_6$ , and (b)  $K_{3,3}$

### 3 Acknowledgements

The author would like to thank David Grant and Bill Macready for enlightening discussions. We would also like to thank Joan and Anthony Geramita, and John Coury for helpful comments.

### References

- [1] M. H. S. Amin, P. J. Love, C. J. S. Truncik, Thermally assisted adiabatic quantum computation (2006).  
URL <http://arxiv.org/abs/cond-mat/0609332>
- [2] M. R. Garey, D. S. Johnson, H. C. So, An application of graph coloring to printed circuit testing, IEEE Transactions on Circuits and Systems 23 (10) (1976) 591–599.
- [3] J. Opatrny, D. Sotteau, Embeddings of complete binary trees into grids and extended grids with total vertex-congestion 1, Discrete Applied Mathematics 98 (2000) 237–254.
- [4] Y. B. Lin, Z. Miller, M. Perkel, D. Pritikin, I. H. Sudborough, Expansion of layouts of complete binary trees into grids, Discrete Applied Mathematics 131 (3) (2003) 611–642.
- [5] F. Sang, I. Sudborough, Embedding large meshes into small ones, IEEE International Symposium on Circuits and Systems (1990) 323–326.
- [6] H. de Fraysseix, J. Pach, R. Pollack, How to draw a planar graph on a grid, Combinatorica 10 (1990) 41–51.
- [7] W. Schnyder, Embedding planar graphs on the grid, in: SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1990, pp. 138–148.